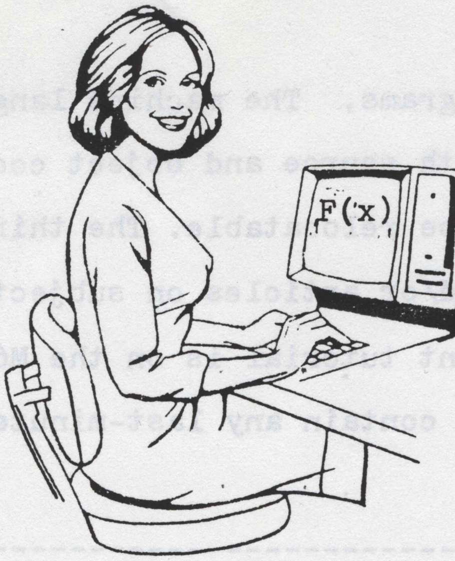
F(X) Software Presents ...



***** The F(x) Journal - For Imagination Machine Users October 1982 *****

The first issue is finally here ! We wish to apologize for the lengthy delay; however, the delay could not be avoided because we had to wait to insure that we would have enough subscribers.

First, let me tell you who we are. The chief writer is yours truly, Roland De Graaf. The supporting writers are Todd Van Klavern and Jay Zuverink.

In case you haven't noticed, this newsletter is intended to be somewhat informal and easy to understand.

In general, this newsletter will four sections. The first section will be a question and answer section. If you include a self-addressed stamped envelope with your questions, you will receive a personal reply, as well as having the question and answer appear in the newsletter. The second section will contain BASIC and/or machine language programs. The machine language listings will usually give both source and object code. The code will generally also be relocatable. The third part will contain tutorials and/or articles on subjects that you requested. The current tutorial is on the M6800 CPU. The fourth part will contain any last-minute information and developments.

SECTION I - Question and Answer

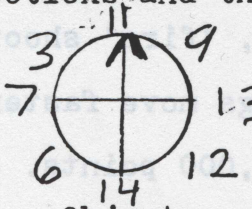
Q. Where do I send my questions ?

A. Send them to : F(x) Software Co.
4246 Elisabeth Ave
Holland, MI
49423

SECTION II - Programs

<u>Eight Direction Joystick Software Modification</u>

The joystick controllers included with the IM-1 leave a little bit to be desired. One of the flaws is the lack of diagonal control. Until now. The following machine language routine will read the joysticks and put values in certain memory locations. The value will indicate the direction. Note that this routine reads <u>both</u> joysticks and only the joysticks, not the keypads. To use the routine, simply key it in using the machine language monitor (the routine can be put into a dummy REM statement), call it up with BASIC CALL instruction when necessary, then read memory locations $01A0 and $01A1 with the BASIC PEEK instruction. (The $ means hexadecimal.) $01A0 equals 416_{ten} and $01A1 euals 417_{ten}. The diagram below shows the directions and the associated values.



To read right joystick, use PEEK(417)
To read left joystick, use PEEK(416)

```
Object      Source code     Comments

B6 81 94    LDAA $8194      Set-up PIA port
84 F0       ANDA #$F0
8A 02       ORAA #$02
B7 81 94    STAA $8194
B6 81 92    LDAA $8192      Read right joystick
84 0F       ANDA #$0F       & get needed bits
B7 01 A1    STAA $01A1      Store in Right loc.
B6 81 92    LDAA $8192      Read left joystick
84 F0       ANDA #$F0       get needed bits
48          ASLA            shift into least
48          ASLA            significant positions
48          ASLA
48          ASLA
B7 01 A0    STAA $01A0      Store in Left loc.
39          RTS
```

Because the joysticks were not made to operate diagonally, they will require some practice. Please, by all means, use this routine in your own programs. This routine could liven up games and simulations by providing more control.

"SPACE STORM" - Hi-res space game

In Space Storm, you are being bombarded by meteors, space rocks, alien ships, stars, and, of course, space junk. This program, written partially in BASIC and partially in machine language. The machine language is held in DATA statements and is POKEd into memory at RUN-time. The object of Space Storm is to shoot the meteors, rocks, alien ships, etc. before they ram you ship. Points are awarded as follows: 20 - meteor, 40 - rock, 60 - alien ship, 80 - space junk, 100 - star. This game uses the left joy stick. Controls are : left and right as usual, 'fire' shoots laser. Note that as the game progresses, things move faster and faster. A bonus ship is awarded at 10,000 points.

Here it is:

```
0 POKE 24578,38: GOTO 10
1 POKE C-1,64: POKE C,65: POKE C+1,66:RETURN
2 POKE C-1,S: POKE C,S: POKE C+1,S: RETURN
3 POKE LC,67: RETURN
4 POKE LC,S: RETURN
5 POKE C-1,B: POKE C,B: POKE C+1,B: RETURN
7 TS=SC: FOR E=4 TO 0 STEP-1: D=INT(TS/10↑E): POKE 14-E,
D+9: TS=TS-D*10↑E: NEXT E: RETURN
8 R=INT(R*RND(R))+1:RETURN
9 CALL 17046: POKE 40960,2: POKE 40961,0: RETURN
10 I=0:C=0:LC=0:E=0:T=0:D=0:S=83:B=84:SC=0:LF=0:L=0:DIM K$(1)
11 FOR I=46000 TO 46023:READ D
12 POKE I,D: NEXT I
13 FOR I=46030 TO 46043: READ D
14 POKE I,D: NEXT I
15 FOR I=46050 TO 46067: READ D
16 POKE I,D: NEXT I
17 FOR I=47000 TO 47127:R=18: GOSUB 8:IF R>5 THEN R=16
18 POKE I,R+67: NEXT I
20 GOSUB 9: PRINT TAB(11);"SPACE STORM";SPC(10)
30 PRINT TAB(11);"----- -----";SPC(10)
40 PRINT
50 PRINT TAB(10);"HIGH SCORE: ";HS
60 PRINT
70 PRINT TAB(10);"GAME SCORE: ";SC
90 PRINT
100 PRINT TAB(1);"PRESS THE (SPACE-BAR) TO START";
105 R=RND(R)
110 IF KEY$(0)<>CHR$(32) GOTO 105
120 GOSUB 9: POKE 40960,3: POKE 40961,128
130 PRINT"SETTING UP HI-RES SHAPES ..."
140 FOR I=512 TO 815
150 READ D: POKE I,D
160 NEXT I
170 FOR I=816 TO 831: POKE I,0: NEXT I
180 FOR I=832 TO 847: READ D: POKE I,D
190 NEXT I
```

```
200 SC=0:EF=0
210 FOR I=0 TO 31: POKE I,S: NEXT I
215 POKE 8193,60: POKE 8194,222
220 POKE 19,0: POKE 20,1: POKE 21,2
230 POKE 23,12
240 CALL 46030
250 D=PEEK(23): IF D=9 GOTO 650
260 D=D-1: POKE 23,D
265 T=0:L=20
270 C=368
280 GOSUB 7: GOSUB 1
300 FOR I=1 TO L
310 K$=KEY$(2): IF K$<>"" GOTO 450
320 IF K$="!" GOTO 360
330 IF K$="W" GOTO 380
340 IF K$="E" GOTO 420
350 GOTO 450
360 LF=1:LC=C-32: GOTO 450
380 IF C-2<352 GOTO 500
390 IF PEEK(C-2)<>S GOTO 620
400 POKE C+1,S:C=C-1: GOSUB 1
410 GOTO 450
420 IF C+2>383 GOTO 500
430 IF PEEK(C+2)<>S GOTO 610
440 POKE C-1,S:C=C+1: GOSUB 1
450 IF LF=0 GOTO 500
460 GOSUB 3:GOSUB 4
470 LC=LC-32: IF LC<32 THEN LF=0: GOTO 500
480 IF PEEK(LC)<>S GOTO 600
490 GOSUB 3: GOSUB 4
500 NEXT I
530 IF PEEK(C-31)<>S GOTO 580
540 IF PEEK(C-32)<> S GOTO 580
550 IF PEEK(C-33)<> S GOTO 580
560 CALL 46000
565 GOSUB 1
570 R=63: GOSUB 8:D=47000+R: POKE 390,D/256:POKE 391,D
575 CALL 46050
577 T=T+1: IF T>50 THEN L=10
578 IF T>100 THEN L=4
579 GOTO 300
580 CALL 46000
590 GOTO 630
600 SC=SC+(PEEK(LC)-67)*20: POKE LC,20: MUSIC"*1":GOSUB 7:
POKE LC,S: MUSIC"/1":LF=0
602 IF SC>=10000 IF EF=0 THEN EF=1: POKE 23,PEEK(23)+1:
MUSIC"*7 *7 *7"
605 GOTO 500
610 GOSUB 2:C=C+1:GOTO 630
620 GOSUB 2:C=C-1
```

```
630 GOSUB 5: MUSIC"/7 /6 /5 /4 /3 /2 /1": FOR I=1 TO 100:
NEXT I
640 GOTO 240
650 GOSUB 9: POKE 8194,30
660 IF SC>HS THEN HS=SC
790 RESTORE : FOR I=1 TO 56: READ D: NEXT I: GOTO 20
800 DATA 206,1,95,198,10,247,1,160,198,32,166,0,167,32,9,
90,38,248,122,1,160,38,241,57
810 DATA 206,0,32,134,83,167,0,8,140,1,128,38,248,57
815 DATA 206,0,32,255,160,41,254,1,134,255,160,43,198,32,
189,119,0,57
820 DATA 0,0,0,0,0,0,0,0,0,0,1,3,7,15,0,0
830 DATA 24,24,24,60,60,60,60,126,126,255,255,255,255,255,
0,0
840 DATA 0,0,0,0,0,0,0,0,0,0,128,192,224,240,0,0
850 DATA 24,24,24,24,24,24,24,24,24,24,24,24,24,24,24,24
860 DATA 56,68,130,129,129,129,129,129,130,68,36,36,56,0,0,0
870 DATA 0,0,0,0,28,34,66,66,66,34,20,8,0,0,0,0
880 DATA 0,66,66,102,102,126,126,126,126,126,126,60,60,
24,24,24
890 DATA 0,0,0,32,34,36,84,42,4,20,18,32,18,40,64,0
900 DATA 0,0,0,0,0,90,60,126,126,60,90,0,0,0,0,0
910 DATA 0,0,0,60,36,36,36,36,36,36,36,60,0,0,0,0
920 DATA 0,0,0,8,8,8,8,8,8,8,8,8,0,0,0,0
930 DATA 0,0,0,60,4,4,4,60,32,32,32,60,0,0,0,0
940 DATA 0,0,0,60,8,8,8,28,8,8,8,60,0,0,0,0
950 DATA 0,0,0,36,36,36,36,36,60,4,4,4,0,0,0,0
960 DATA 0,0,0,60,32,32,32,60,4,4,4,60,0,0,0,0
970 DATA 0,0,0,60,32,32,32,60,36,36,36,60,0,0,0,0
980 DATA 0,0,0,60,4,4,4,4,4,4,4,4,0,0,0,0
990 DATA 0,0,0,60,36,36,36,60,36,36,36,60,0,0,0,0
1000 DATA 0,0,0,60,36,36,36,60,4,4,4,60,0,0,0,0
1010 DATA 36,1,136,32,9,166,88,58,89,50,140,33,20,128,1,16
9000 END
```

The program is 111 lines long and occupies 2687 bytes. One final note about the program: You are allowed only one laser shot on the screen at any one time, however, if you fire the laser while another is in flight, the older one will disappear and a new one will start. This feature was added so that accidental shooting wouldn't be a problem.

Have Fun !!

SECTION III

6800: Your Microprocessor-Part 1 of 2

In case you haven't noticed, the 40-pin chip which does all the arithmetic and data manipulation in the IM is the M6800 microprocessor chip, produced by Motorola. The 6800 was introduced around 1972, and was expected to compete with the 8080 produced by Intel. As time went by, both chips flourished. Today, few new systems are designed around these chips. This is mainly because of two other chips, the 6502 by Mostek and the Z-80 by Zilog. Enough history.

The 6800, which is still considered to be a very powerful chip, is fairly easy to program compared with other microprocessors.

The 6800 contains 6 accessible registers: the two accumulators, A and B, the index register, X, the stack pointer, SP, the program counter, PC, and the condition codes register, CCR. A,B, and CCR are 8 bit registers; while X, SP, and PC are 16 bit registers.

The two accumulators, A and B, are used for performing arithmetic and storing data. The CCR shows status afte certain events occur, such as if an addition produced an overflow or a negative or positive number. Next month we will cover the CCR and A and B in greater detail.

The remaining registers, X, SP, and PC are 16 bits long and are used for pointing at memory locations; in other words, they most often contain memory addresses. Again, these registers will be covered in greater detail next month.

Programming a microprocessor is no more than controlled manipulation of the registers and associated memory and I/O devices. The flexible instruction set of the 6800, when utilized correctly with the needed support equipment, can instruct the microprocessor to accomplish almost any task.

(continued next month)

SECTION IV : Last minute info & Abstract thoughts

1 If you would like to advertise in this newsletter, write us. Advertising for subscribers is free.

2 The F(x) Software Catalog should be out sometime in November.

3 We welcome your ideas, suggestions and criticisms; write us.

4 Although we do not rely on subscriber programs to make up the bulk of our newsletter as other mewsletters do, we welcome your programs that you would like to share.

5 Next Month : TRIG functions, 6800-part 2, and much more

99 END